UNITED STATES PATENT AND TRADEMARK OFFICE

A

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/085,455	02/27/2002	Motohiro Kawahito	JP920000420US1	1801

7590 07/26/2005

IBM CORPORATION
INTELLECTUAL PROPERTY LAW DEPT.
P.O. BOX 218 - 39-254
YORKTOWN HEIGHTS, NY 10598

EXAMINER	
PHAM, CHRYSTINE	
ART UNIT	**PAPER NUMBER**
2192	

DATE MAILED: 07/26/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	10/085,455	KAWAHITO ET AL.	
	Examiner	Art Unit	
	Chrystine Pham	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *26 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-16* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-16* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is responsive to Amendment filed on April 26th 2005. Claim 2 has been amended. Claims 1-16 are presented for examination.

Response to Amendment

2. In view of the amendment to claim 2 to include proper reference to base claim 1, objection to claim is hereby withdrawn.

Response to Arguments

3. Applicants' arguments filed April 26th 2005 have been fully considered but they are not persuasive.

First, regarding pages 9-12 of Applicants' remarks, in which Applicants describe their invention as to contrast it against that of Faiman, it is noted that Applicants repeatedly characterize Faiman's compiler as "prior art static compiler" while emphasizing that the compiler of their invention is a "dynamic compiler" that is capable of "generating a path for fixing the parameter dynamically" (emphasis added). It is submitted that, neither the "dynamic compiler" (page 10, last 3 lines of first full paragraph) nor "generating a path for fixing the parameter dynamically" (page 11, lines 2-3) can be found in the pending claims. Thus, Applicants' argument with respect to the static/dynamic compiler feature is deemed irrelevant to their invention as claimed, and the invention as claimed does not distinguish over the cited prior art.

Second, the Applicants contend that "Faiman does not perform any analysis to determine whether the execution speed of a program can be increased by fixing, in a specific state, a parameter for a predetermined command in said program." (pages 12-13). The Applicants further rely on col.22:12-16 to assert that "Faiman operates on the assumption that eliminating any fetch operations will automatically result in reduced speed" (page 13 lines 3-5). It is submitted that, this assertion is merely a characterization of Faiman's invention, not to mention an improper one, since Faiman's invention clearly aims to optimize (i.e., increase), not reduce, the execution speed of the program. As has been established in the previous Office Action, Faiman discloses an optimizing technique, which is performed during compiling, in which the program is rewritten (i.e., fixed or modified) to optimize (i.e., increase) the execution speed of the program (see at least col.2:28-35). In col.4:15-30, and col.22:6-35, Faiman specifically discloses the KFOLD routine, as part of the optimizer 26 (i.e., compiler), which detects the occurrences in the program, where expressions (see at least *source code listing, A, B, A+B* col.22:6-35) can be reduced to a constant which is calculated at compile time. In this context, an expression in Faiman's program clearly anticipates to the claimed "parameter for a predetermined command in said program" and Faiman's reducing said expression to a calculated constant clearly anticipates "fixing, in a specific state, a parameter for a predetermined command in said program". Furthermore, since the KFOLD

routine is a constant expression evaluation routine performed during compiling, the program has to be analyzed in order to detect/evaluate constant expressions and calculate the constants thereof, and whether the execution speed of the program can be increased is determined by the outcome of the KFOLD routine, it is inherent that "an analysis is performed to determine whether the execution speed of said program can be increased".

Third, the Applicants contend that Faiman does not anticipate "the claimed steps of employing analysis results to generate a path along which a parameter is fixed in a specific state, let alone of obtaining statistical data for the appearance frequency of each available state for the program parameter" (page 13). It is submitted that, in col.22:6-20, Faiman specifically discloses calculating the constants for the constant expressions during compile time, and folding (i.e., incorporating) the calculated constants in the object code image to be executed. It is further submitted that constant expressions to be evaluated, and replaced by constants are part of the intermediate language graph 55, which consists of tuples in ordered sequences within blocks, where a block is part of the code that begins with a routine or label and ends in a branch, and the interlinked blocks make up the graph (i.e., flow graph) (see at least col.3:15-35). It is further submitted that Faiman discloses executing the generated code (for the expressions) in the same order (i.e, path) that the expressions occurred in the intermediate language graph (see at least col.27:30-col.28:30). It is clear that the intermediate language graph anticipates different branches (i.e., paths) of the

program. And since, each expression to be evaluated by the KFOLD routine is part of a branch, and the executable code (i.e., folded constant) to be generated for said expression follows the same order (i.e., path) that the expression occurred in the graph, it is inherent that "a path is generated along which said parameter of said predetermined command is fixed in said specific state".

With respect to the claimed "obtaining statistical data for the appearance frequency of each available state for the program parameter", in col.18:40-col.19:50, Faiman discloses analyzing induction variables and inductive expressions. An induction variable is a variable that increments or decrements by the same amount (i.e., constant) each time it is executed (see at least *X* col.19:15-50). And an inductive expression is an expression that can be computed as a linear function of an induction variable (see at least *inductive expression I*4* col.19:15-50). Faiman discloses replacing the inductive expression 4*I with I2, which has the value of 4 (i.e., constant). Without "obtaining statistical data for the appearance frequency of each available state", that is to say, without a analyzing the code to detect which variables increment/decrement by the same amount and executed at most once in every "complete trip" trip through the loop, it is impossible identify and eliminate the variables (i.e., replace inductive expressions with constants) in order to optimize execution speed of the program.

In view of the fore going discussion, rejection of claims under 35 USC 102(b) is considered proper and maintained.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

5. Claims 1-16 are rejected under 35 U.S.C. 102(b) as being anticipated by Faiman, Jr. (USP 5836014), hereinafter, *Faiman.*

Claim 1

Faiman teaches a program optimization method (e.g., see *optimizing 26* FIG.1 & associated text) for translating, into machine code (e.g., see *object code 23, executable image, target computer 25* FIG.1 & associated text; see *object code images, machine language* col.2:49-55), source code for a program written in a programming language (e.g., see *source code 21* FIG.1 & associated text; see *compiler front end* col.2:5-16), and for optimizing said program (e.g., see *optimizing 26* FIG.1 & associated text) comprising the steps of:

- performing an analysis to determine whether the execution speed of said program can be increased (e.g., see *optimize speed of execution* col.2:28-35) by fixing, in a specific state (e.g., see *K-folding, KFOD routine, constant*

col.4:14-21), a parameter for a predetermined command in said program (e.g., see *expressions* col.4:14-21; col.22:29-34); and

- employing results of said analysis for the generation, in said program, of a path along which said parameter of said predetermined command is fixed in said specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 2

The rejection of base claim 1 is incorporated. *Faiman* further teaches wherein said step of generating a path includes the steps of:

- executing said program and obtaining statistical data for the appearance frequency of each available state (e.g., see *analyzing induction variables* col.3:65-col.4:5; see *detection of induction variables* col.18:40-67) wherein, according to said results of said analysis, said parameter of said predetermined command may be set (e.g., see *inductive expressions, multiplications, additions* col.4:1-10); and
- employing said obtained statistical data (e.g., see *inductive expressions* col.18:64-col.19:50) to generate said path.

Claim 3

Faiman teaches a program optimization method (e.g., see *optimizing 26* FIG.1 & associated text) for translating, into machine code (e.g., see *object code 23, executable image, target computer 25* FIG.1 & associated text; see *object code images, machine language* col.2:49-55), the source code for a program written in a programming language (e.g., see *source code 21* FIG.1 & associated text; see *compiler front end* col.2:5-16), and for optimizing said program (e.g., see *optimizing 26* FIG.1 & associated text) comprising the steps of:

- executing a program to obtain statistical data for an appearance frequency of each available state (e.g., see *analyzing induction variables* col.3:65-col.4:5; see *detection of induction variables* col.18:40-67) in which a parameter of a predetermined command in said program may be set (e.g., see *inductive expressions, multiplications, additions* col.4:1-10); and
- employing said obtained statistical data (e.g., see *inductive expressions* col.18:64-col.19:50) to generate a machine language program that includes, as the compiling results, a path (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35) along which said parameter of said predetermined command (e.g., see *expressions* col.4:14-21; col.22:29-34) is fixed in a specific state (e.g., see *K-folding, KFOD routine, constant* col.4:14-21).

Claim 4

The rejection of base claim 3 is incorporated. *Faiman* further teaches comprising a step of: generating a machine language program that does not include, as a compiling result, a path along which said parameter of said predetermined command is fixed in a specific state (e.g., see *conditional stores* col.20:8-14).

Claim 5

Faiman teaches a program optimization method (e.g., see *optimizing 26* FIG.1 & associated text) for translating, into machine code (e.g., see *object code 23, executable image, target computer 25* FIG.1 & associated text; see *object code images, machine language* col.2:49-55), the source code for a program written in an object-oriented programming language (e.g., see *source code 21, compiler front end 20* FIG.1 & associated text; see *front end, C++* col.2:65-col.3:10; see *compiler front end* col.2:5-16),

and for optimizing said program (e.g., see *optimizing 26* FIG.1 & associated text) comprising the steps of:

- detecting one command, of the commands in said program, for which a method call destination can be identified (e.g., see *tuples, routine call, procedure calls* col.14:47-65), and for which the processing speed can be increased by identifying said method call destination (e.g., see *tuples* col.12:59-65; see *flow graphs, procedure calls, variable, memory locations* col.13:43-55; ; see *optimize speed of execution, flow graph* col.2:28-34); and

- generating a path (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35) wherefor said method call destination for said detected command is limited in order to increase the processing speed of said command (e.g., see *Restrictions, tuple* col.55:53-66).

Claim 6

Faiman teaches a program optimization method (e.g., see *optimizing 26* FIG.1 & associated text) for translating, into machine code (e.g., see *object code 23, executable image, target computer 25* FIG.1 & associated text; see *object code images, machine language* col.2:49-55), the source code for a program written in a programming language (e.g., see *source code 21* FIG.1 & associated text; see *compiler front end* col.2:5-16), and for optimizing said program (e.g., see *optimizing 26* FIG.1 & associated text) comprising the steps of:

- detecting one command, of the commands in said program (e.g., see *expressions* col.4:14-21; col.22:29-34), for which a variable can be limited to a predetermined constant value (e.g., see *K-folding, KFOD routine, constant* col.4:14-21), and for which the processing speed can be increased by limiting said variable to said constant value (e.g., see *optimize speed of execution* col.2:28-35); and

- generating a path along which said constant value of said variable of said detected command is fixed (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 7

Faiman teaches a compiler (e.g., see *compiler front end 20* FIG.1 & associated text) for translating into machine code (e.g., see *object code 23, executable image, target computer 25* FIG.1 & associated text; see *object code images, machine language* col.2:49-55) the source code for a program written in a programming language (e.g., see *source code 21* FIG.1 & associated text; see *compiler front end* col.2:5-16), and for optimizing the resultant program (e.g., see *optimizing 26* FIG.1 & associated text) comprising:

- an impact analysis unit for performing an analysis to determine how much (e.g., see *effects 42, dependencies 43* FIG.4 & associated text) the execution speed of said program can be increased (e.g., see *optimize speed of execution* col.2:28-35) by fixing, in a specific state (e.g., see *K-folding, KFOD routine, constant* col.4:14-21), a parameter of a predetermined command in said program (e.g., see *expressions* col.4:14-21; col.22:29-34); and
- a specialization unit for employing the analysis results obtained by said impact analysis unit to generate, in said program, a specialized path along which said

parameter of said predetermined command is fixed in said specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 8

The rejection of base claim 7 is incorporated. *Faiman* further teaches:

- a data specialization selector for, when said program is executed, obtaining statistical data for the appearance frequency of each state obtained by said impact analysis unit (e.g., see *analyzing induction variables* col.3:65-col.4:5; see *detection of induction variables* col.18:40-67), and for determining the state in which said parameter of said predetermined command is to be set (e.g., see *inductive expressions, multiplications, additions* col.4:1-10),
- wherein said specialization unit generates a specialized path along which said parameter of said predetermined command is fixed in a state determined by said data specialization selector (e.g., see *inductive expressions* col.18:64-col.19:50).

Claim 9

The rejection of base claim 8 is incorporated. *Faiman* further teaches wherein, in accordance with the state of said program at execution, said specialization unit generates, in said program, a branching process for selectively performing a

specialized path and an unspecialized path (e.g., see *conditional branch, cases* col.23:20-40); and wherein, while taking into account a delay due to the insertion of said branching process (e.g., see *Delayed Actions, passes* col.25:30-40), said data specialization selector determines a state in which said parameter of said predetermined command is fixed (e.g., see *inductive expressions* col.18:64-col.19:50).

Claim 10

Faiman teaches a computer (e.g., see FIGS.1, 2 & associated text) comprising:

- an input device for receiving source code for a program (e.g., see *source code 21, compiler front end 20, shell 11* FIG.1 & associated text; col.6:14-21);
- a compiler (e.g., *compiler front end 20* FIG.1 & associated text) for translating said source code to compile said program (e.g., see *compiler front end* col.2:5-16) and for converting said compiled program into machine language code (e.g., see *object code 23, executable image* FIG.1 & associated text; see *object code images, machine language* col.2:49-55); and
- a processor for executing said machine language code (e.g., see *CPU 14* FIG.2 & associated text; see *target computer 25* FIG.1 & associated text),
- wherein said compiler includes
- means for performing an analysis to determine whether the execution speed of said program can be improved (e.g., see *optimize speed of execution* col.2:28-35) by fixing in a specific state (e.g., see *K-folding, KFOD routine, constant*

col.4:14-21) a parameter of a predetermined command in said program (e.g., see *expressions* col.4:14-21; col.22:29-34), and

- means for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said specific state and for compiling said program (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35), and
- wherein said compiler outputs, as the compiled results, said machine language code that includes said path along which the state of said parameter is fixed (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 11

Faiman teaches a computer (e.g., see FIGS.1, 2 & associated text) comprising:

- an input device, for receiving source code for a program (e.g., see *source code 21, compiler front end 20, shell 11* FIG.1 & associated text; col.6:14-21);
- a compiler (e.g., *compiler front end 20* FIG.1 & associated text), for translating said source code to compile said program (e.g., see *compiler front end* col.2:5-16) and for converting said compiled program into machine language code (e.g.,

see *object code 23, executable image* FIG.1 & associated text; see *object code images, machine language* col.2:49-55); and

- a processor, for executing said machine language code (e.g., see *CPU 14* FIG.2 & associated text; see *target computer 25* FIG.1 & associated text),
- wherein said compiler includes
- means for obtaining statistical data for the appearance frequency of each available state wherein a parameter for a predetermined command in said program may be set when said program is executed (e.g., see *analyzing induction variables* col.3:65-col.4:5; see *detection of induction variables* col.18:40-67), and for employing said statistical data to determine a state in which said parameter of said predetermined command is to be fixed (e.g., see *inductive expressions, multiplications, additions* col.4:1-10), and
- means for generating a specialized path along which said parameter of said predetermined command is fixed in said determined state, and for compiling said program (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35), and
- wherein said compiler outputs, as the compiled results, said program as said machine language code that includes said specialized path (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 12

The rejection of base claim 11 is incorporated. *Faiman* further teaches comprising: said compiler further includes means for compiling said program without generating a specialized path, wherein, when said state of said parameter to be fixed can not be determined, said means for determining the state of said parameter of said predetermined command outputs, as compiled results, said program in said machine language code, which is generated by said means for compiling said program without generating said specialized path, that does not include said specialized path (e.g., see *conditional stores* col.20:8-14).

Claim 13

Faiman teaches a support program, for controlling a computer to support generation of a program (e.g., see *machine code*, *constant expression evaluation routine* col.23:32-35), which permits said computer to perform:

- a function for performing an analysis to determine whether the execution speed of said program can be increased (e.g., see *optimize speed of execution* col.2:28-35) by fixing a parameter of a predetermined command (e.g., see *expressions* col.4:14-21; col.22:29-34) of said computer program in a specific state (e.g., see *K-folding, KFOD routine*, *constant* col.4:14-21); and
- a function for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said

specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 14

Faiman teaches a support program, for controlling a computer to support generation of a program (e.g., see *machine code, constant expression evaluation routine* col.23:32-35), which permits said computer to perform:

- a function for executing said program and obtaining statistical data for the appearance frequency of each available state (e.g., see *analyzing induction variables*.col.3:65-col.4:5; see *detection of induction variables* col.18:40-67) wherein said parameter of said predetermined command of said program may be set (e.g., see *inductive expressions, multiplications, additions* col.4:1-10); and
- a function for generating in said program, based on said statistical data, a path along which said parameter of said predetermined command is fixed in said specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35; *inductive expressions* col.18:64-col.19:50).

Claim 15

Faiman teaches a storage medium (e.g., see *memory 15, 17* FIG.2 & associated text) on which input means of a computer stores a computer-readable support program (e.g., see *machine code, constant expression evaluation routine* col.23:32-35), for controlling said computer to support generation of a program, that permits said computer to perform:

- a function for performing an analysis to determine whether the execution speed of said program can be increased (e.g., see *optimize speed of execution* col.2:28-35) by fixing a parameter of a predetermined command (e.g., see *expressions* col.4:14-21; col.22:29-34) of said computer program in a specific state (e.g., see *K-folding, KFOD routine, constant* col.4:14-21); and
- a function for generating in said program, based on the analysis results, a path along which said parameter of said predetermined command is fixed in said specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35).

Claim 16

Faiman teaches a storage medium (e.g., see *memory 15, 17* FIG.2 & associated text) on which input means of a computer stores a computer-readable support program (e.g., see *machine code, constant expression evaluation routine* col.23:32-35), for controlling said computer to support generation of a program, that permits said computer to perform:

- a function for executing said program and obtaining statistical data for the appearance frequency of each available state (e.g., see *analyzing induction variables* col.3:65-col.4:5; see *detection of induction variables* col.18:40-67) wherein said parameter of said predetermined command of said program may be set (e.g., see *inductive expressions, multiplications, additions* col.4:1-10); and
- a function for generating in said program, based on said statistical data, a path along which said parameter of said predetermined command is fixed in said specific state (e.g., see *constant expression evaluation routine, runtime, object code image, Kfold routine* col.22:6-20; see *generating code* col.22:67-col.23:8; see *machine code, constant expression evaluation routine* col.23:32-35; *inductive expressions* col.18:64-col.19:50).

Conclusion

6. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be

calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the mailing date of this final action.

7. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chrystine Pham whose telephone number is 571-272-3702. The examiner can normally be reached on Mon-Fri, 8:30am-5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

CP
July 20, 2005

Chameli C. Das
CHAMELI C. DAS
PRIMARY EXAMINER
7/25/05